CMPT 135

Sample Midterm Exam 1 **Solutions**

Based on Spring 2021 midterm

This is **50 minute closed book exam**: no notes, books, computers, calculators, electronic devices, etc. are permitted. Do not speak to any other students during their exam or look at their work. If you have a question, please remain seated and **raise your hand** and a proctor will come to you.

Basic C++

a) Write a function called `is_vowel(c)` that returns `true` when the character `c` is a vowel, and `false` otherwise. For this question a vowel is considered to be exactly one of the 5 lowercase letters *a*, *e*, *i*, *o*, or *u*.

Sample Solution:

```
bool is_vowel(char c) {
    switch (c) {   // could also use a big if-else-if structure
        case 'a': case 'e':
        case 'i': case 'o':
        case 'u': return true;
        default : return false;
    }
}
```

b) Write a function called `remove_vowels(s)` that takes a C++ string `s` as input and returns a new string that is the same as `s` except that all vowels have been removed. For example, `remove_vowels("Apple sauce")` returns the string `"Appl sc"`.

Use `is_vowel` from the previous question to determine when a character is a vowel. Assume `string` is `#include`-ed, but no other files are `#include`-ed.

Sample Solution:

```
// returns a string
string remove_vowels(const string& s) {  // s passed by constant reference
    string result;
    for(char c : s) {  // process string s one character at time
        if (!is_vowel(c)) result += c;  // only add non-vowels to result
    }
    return result;
}
```

Pointers and Arrays

(10 marks) Inside a `void` function named `pointers_and_arrays()`, write C++ code that:

- creates a new array of 500 `double`s on the free store
- sets the entries of this array to be the values from 500 down to 1; that is, the first value of the array is set to 500, the second value to 499, ..., and the last value to 1
- using a loop, reverses the order the elements of the array so they go from 1 to 500
- using a loop *different* than the previous one, prints all the values of the array to `cout`, one number per line, but *without* using the `[]`-notation anywhere in the loop (you can use `[]`-brackets in any other part of this question, but just not here)
- has no memory leaks

Solution:

```
void pointers_and_arrays() {
      double* arr = new double[500]; // create array (2 marks)

      for(int i = 0; i < 500; i++)    // init array (2 marks)
         arr[i] = 500 - i;

      int a = 0;                      // reverse array (3 marks)
      int b = 499;

      while (a < b) {
            swap(arr[a], arr[b]);
            a++;
            b--;
      }

     for(int i = 0; i < 500; i++)     // print array (2 marks)
        cout << *(arr + i) << "\n";

     delete[] arr;                    // de-allocate array (1 mark)
}
```

Classes

(20 marks) Write a class called `Product` that stores the name (a `string`) and cost (a `double`) of a store-bought product. Your class must have these features:

- All member variables are **private**.
- All methods are **public**.
- A **default constructor** that uses **member initialization** to set the product's name to "none", cost to -1, and prints the message "object created" to `cout`.
- A **constructor** that uses an **initialization list** to set the products name and cost to values passed into the constructor. If the name is an empty string, or if the cost is less than 0, then it should throw an error using `cmpt::error`.
- A **copy constructor** that uses **constructor delegation** to set the product's name and cost to be the same as the name and cost of another passed-in `Product` object.
- A **destructor** that prints the message "object deleted".
- A **getter** that returns the cost of the product.
- A **setter** that sets the cost of the product to be a given `double`. If the given `double` is less than 0, then it should throw an error using `cmpt::error`.
- Define an `==` operator that tests if two `Product` objects have the same name and cost. Importantly, define this `==` *outside* of the `Product` class.
- Write this code neatly, and use good indentation and C++ style.

```
    class Product {
private:
    string name = "none";    // member initialization
    double cost = -1;


public:
    Product() {              // default constructor
        cout << "object created\n";
    }


    Product(const string& n, double c)
    : name(n), cost(c)  // initializer list
    {
        if (name == "" || cost < 0) cmpt::error("bad data");
    }
```

```
    Product(const Product& other)       // copy constructor
    : Product(other.name, other.cost)  // constructor delegation
    { }

    ~Product() {              // destructor
        cout << "object deleted\n";
    }

    // getters
    string get_name() const { return name; }
    double get_cost() const { return cost; }

    // setters
    void set_cost(double c) { cost = c;    }
}; // class Product

bool operator==(const Product& a, const Product& b) {
    return (a.get_name() == b.get_name())
        && (a.get_cost() == b.get_cost());
}
```